1. 文件结构

Data:里面放的是一些配置文件和想定文件之类的

Config:对于KD来说暂时每用

modelInfo:模型信息文件夹

ModelConfig:这里面的文件存储了模型挂载了哪些组件的信息

model\_enum.config:这个文件暂时好像没用上

model\_map.config:模型名跟中文名的对应表

Scenario:存储想定文件，目前写的很简单,keda.xml里面放了一个无人机和几个人，可用作测试

Dll:动态库生成位置

Lib:静态库生成位置

Log:log打印位置

Material:客户端软件素材，KD这里用不上

Projects:VS中创建的工程的位置

Src:代码存放位置

1. 代码结构

Components:组件相关代码ComponentBase为组件基类,生成为lib库,其派生的子类生成为动态库

ComponentBase基类提供以下接口

virtual void Init(TiXmlElement\* unitElement);组件初始化，模型创建该组件时调用，需要重写

virtual void ReadScenario();读取想定信息，引擎生成该组件时调用，需要重写

virtual void PostEvent();往外推送事件，但是现在还没有用上，用上后需要重写

virtual void ReceiveEvent(EventBase \*event);处理事件，引擎主动调用，需要重写

virtual void Run(double t);组件周期运行函数，需要重写

virtual void Destory();模型销毁，暂时没用上

void HandleEvent();模型从组件拿需要推送的事件，暂时没用上

void HandleComponentState();暂时没用上，现在也想不起来这个写了干嘛用的

void SetBasicInfo(Model\_BasicInfo info);实体创建该组件对象时告诉组件模型的基本信息

Model\_Position GetPos();从组件获取坐标信息，实体判断该组件是移动类型组件时调用，当然，这种方式其实是不合理的，但是我没时间改

Models的逻辑同上

ModelBase基类提供以下接口

virtual void Init(TiXmlElement\* unitElement);模型初始化，引擎生成该模型时调用，需要重写

virtual void ReadScenario();读取想定信息，引擎生成该模型时调用，需要重写

virtual void PostEvent();往外推送事件，但是现在还没有用上，用上后需要重写

virtual void ReceiveEvent(EventBase \*event);处理事件，引擎主动调用，需要重写

virtual void Run(double t);模型周期运行函数，需要重写

virtual void Destory();模型销毁，暂时没用上

void SetID(int id);引擎创建模型时给模型设置ID

void GetBasicInfo(Model\_BasicInfo &info);外部获取模型信息接口

void HandleEvent();引擎从模型拿需要推送的事件，暂时没用上

void HandleModelState();暂时没用上

void InitComponent();模型创建该组件时主动调用，初始化组件

void PutEventToComponent();模型将接收到的事件推给组件

Tools:一些通用工具生成的静态库，目前里面之有一个XML工具接口

EngineController:通过界面操作引擎和模型的工具，只是一个工具，不影响模型，可用来测试

MyEngineClient:KD项目无用

1. 引擎如何使用模型
2. 创建模型

参考MyEngine.cpp中的ReadScenario函数，这里写的时引擎读取想定文件，并将模型创建出来，MM指的是模型管理器，用于管理模型的运行

1. 使用模型

模型运行再线程池中，具体的代码参考ModelRunner函数

1. 模型加载组件

在引擎创建模型后调用model->InitComponent()，参考MyEngine.cpp的275行

1. 发送指令

第一步:在MyEngine中调用void PutEvent(EventBase \* event);

第二步：在运行模型时调用model->PutEventToComponent();参考MyEngine.cpp的53行

第三步:从这里就可以调用到组件的ReceiveEvent中

1. 模型和组件怎么交互

ModelRunner函数调用了model->Run();

在模型的基类中执行了运行组件和读取组件返回的数据

参考ModelBase.cpp中的void ModelBase::Run(double t),这里就调用到了组件动态库中